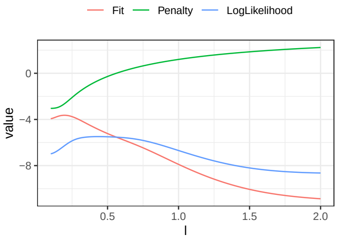
Introduction to Machine Learning

Gaussian Process Training



Learning goals

- Training of GPs via Maximum Likelihood estimation of its hyperparameters
- Computational complexity is governed by matrix inversion of the covariance matrix

TRAINING OF A GAUSSIAN PROCESS

- To make predictions for a regression task by a Gaussian process, one simply needs to perform matrix computations.

- But for this to work out, we assume that the covariance functions is fully given, including all of its hyperparameters.

- A very nice property of GPs is that we can learn the numerical hyperparameters of a selected covariance function directly during GP training.

TRAINING A GP VIA MAXIMUM LIKELIHOOD

Let us assume

$$y = f(\mathbf{x}) + \epsilon,\ \epsilon \sim \mathcal{N}\left(0, \sigma^2\right),$$

where $f(\mathbf{x}) \sim \mathcal{GP}\left(\mathbf{0}, k\left(\mathbf{x}, \mathbf{x}' | \boldsymbol{\theta}\right)\right)$.

Observing $\boldsymbol{y} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{K} + \sigma^2 \boldsymbol{I}\right)$, the marginal log-likelihood (or evidence) is

$$\begin{aligned} \log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) &= \log\left[(2\pi)^{-n/2}\left|\boldsymbol{K}_y\right|^{-1/2} \exp\left(-\frac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}_y^{-1}\boldsymbol{y}\right)\right] \\ &= -\frac{1}{2}\boldsymbol{y}^T \boldsymbol{K}_y^{-1}\boldsymbol{y} - \frac{1}{2}\log\left|\boldsymbol{K}_y\right| - \frac{n}{2}\log 2\pi. \end{aligned}$$

with $\boldsymbol{K}_y := \boldsymbol{K} + \sigma^2 \boldsymbol{I}$ and $\boldsymbol{\theta}$ denoting the hyperparameters (the parameters of the covariance function).

TRAINING A GP VIA MAXIMUM LIKELIHOOD

The three terms of the marginal likelihood have interpretable roles, considering that the model becomes less flexible as the length-scale increases:

- the data fit $-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{K}_y^{-1}\boldsymbol{y}$, which tends to decrease if the length scale increases
- the complexity penalty $-\frac{1}{2}\log|\boldsymbol{K}_y|$, which depends on the covariance function only and which increases with the length-scale, because the model gets less complex with growing length-scale
- a normalization constant $-\frac{n}{2}\log 2\pi$

TRAINING A GP: EXAMPLE

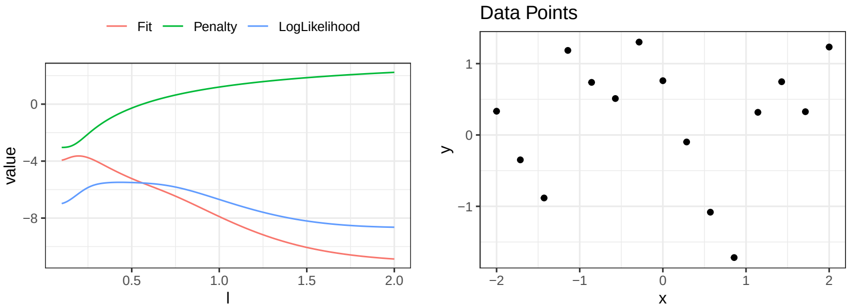
To visualize this, we consider a zero-mean Gaussian process with squared exponential kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2\ell^2}\|\mathbf{x} - \mathbf{x}'\|^2\right),$$

- Recall, the model is smoother and less complex for higher length-scale ℓ.
- We show how the
  - data fit $-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{K}_y^{-1}\boldsymbol{y}$,
  - the complexity penalty $-\frac{1}{2}\log|\boldsymbol{K}_y|$, and
  - the overall value of the marginal likelihood $\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$

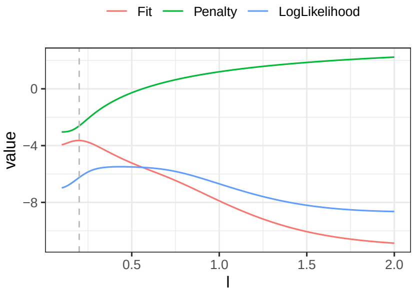
  behave for increasing value of ℓ.

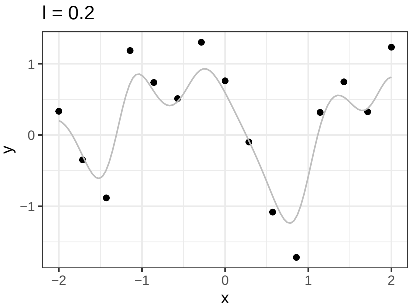
TRAINING A GP: EXAMPLE



The left plot shows how values of the data fit $-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{K}_y^{-1}\boldsymbol{y}$, the complexity penalty $-\frac{1}{2}\log|\boldsymbol{K}_y|$ (high value means less penalization) and the overall marginal likelihood $\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$ behave for increasing values of ℓ.

TRAINING A GP: EXAMPLE

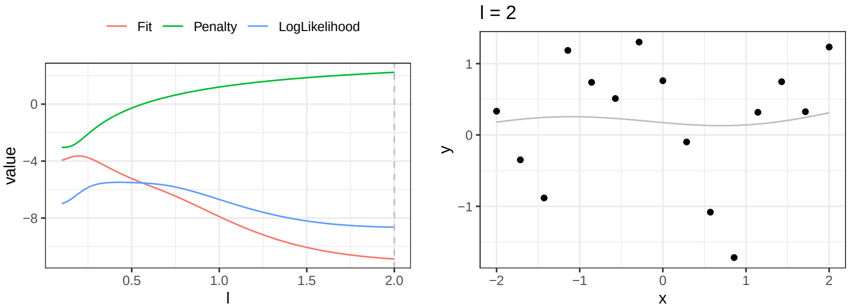




The left plot shows how values of the data fit $-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{K}_y^{-1}\boldsymbol{y}$, the complexity penalty $-\frac{1}{2}\log|\boldsymbol{K}_y|$ (high value means less penalization) and the overall marginal likelihood $\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$ behave for increasing values of ℓ.

A small ℓ results in a good fit, but a high complexity penalty (low $-\frac{1}{2}\log|\boldsymbol{K}_y|$).

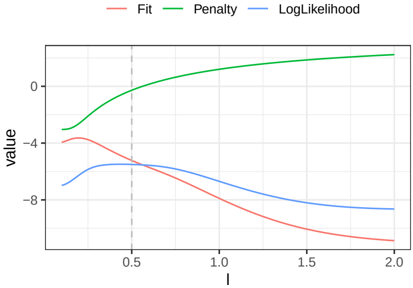
TRAINING A GP: EXAMPLE

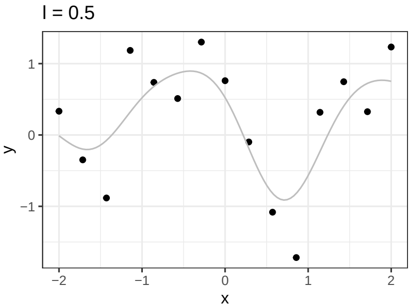


The left plot shows how values of the data fit $-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{K}_y^{-1}\boldsymbol{y}$, the complexity penalty $-\frac{1}{2}\log|\boldsymbol{K}_y|$ (high value means less penalization) and the overall marginal likelihood $\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$ behave for increasing values of ℓ.

A large ℓ results in a poor fit.

TRAINING A GP: EXAMPLE





The left plot shows how values of the data fit $-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{K}_y^{-1}\boldsymbol{y}$, the complexity penalty $-\frac{1}{2}\log|\boldsymbol{K}_y|$ (high value means less penalization) and the overall marginal likelihood $\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$ behave for increasing values of ℓ.

The maximizer of the log-likelihood, $\ell = 0.5$, balances complexity and fit.

TRAINING A GP VIA MAXIMUM LIKELIHOOD

To set the hyperparameters by maximizing the marginal likelihood, we seek the partial derivatives w.r.t. the hyperparameters

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} \log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) &= \frac{\partial}{\partial \theta_j} \left(-\frac{1}{2} \boldsymbol{y}^T \boldsymbol{K}_y^{-1} \boldsymbol{y} - \frac{1}{2} \log |\boldsymbol{K}_y| - \frac{n}{2} \log 2\pi \right) \\
&= \frac{1}{2} \mathbf{y}^\top \boldsymbol{K}^{-1} \frac{\partial \boldsymbol{K}}{\partial \theta_j} \boldsymbol{K}^{-1} \mathbf{y} - \frac{1}{2} \mathrm{tr} \left(\boldsymbol{K}^{-1} \frac{\partial \boldsymbol{K}}{\partial \boldsymbol{\theta}} \right) \\
&= \frac{1}{2} \mathrm{tr} \left((\boldsymbol{K}^{-1} \boldsymbol{y} \boldsymbol{y}^T \boldsymbol{K}^{-1} - \boldsymbol{K}^{-1}) \frac{\partial \boldsymbol{K}}{\partial \theta_j} \right)
\end{aligned}
$$

using $\frac{\partial}{\partial \theta_j} \boldsymbol{K}^{-1} = -\boldsymbol{K}^{-1} \frac{\partial \boldsymbol{K}}{\partial \theta_j} \boldsymbol{K}^{-1}$ and $\frac{\partial}{\partial \boldsymbol{\theta}} \log |\boldsymbol{K}| = \mathrm{tr} \left(\boldsymbol{K}^{-1} \frac{\partial \boldsymbol{K}}{\partial \boldsymbol{\theta}} \right)$.

TRAINING A GP VIA MAXIMUM LIKELIHOOD

- The complexity and the runtime of training a Gaussian process is dominated by the computational task of inverting $\boldsymbol{K}$ - or let's rather say for decomposing it.
- Standard methods require $\mathcal{O}(n^3)$ time (!) for this.
- Once $\boldsymbol{K}^{-1}$ - or rather the decomposition -is known, the computation of the partial derivatives requires only $\mathcal{O}(n^2)$ time per hyperparameter.
- Thus, the computational overhead of computing derivatives is small, so using a gradient based optimizer is advantageous.

TRAINING A GP VIA MAXIMUM LIKELIHOOD

Workarounds to make GP estimation feasible for big data include:

- using kernels that yield sparse $\boldsymbol{K}$: cheaper to invert.
- subsampling the data to estimate θ: $\mathcal{O}(m^3)$ for subset of size m.
- combining estimates on different subsets of size m:
  Bayesian committee, $\mathcal{O}(nm^2)$.
- using low-rank approximations of $\boldsymbol{K}$ by using only a representative subset ("inducing points") of m training data $\boldsymbol{X}_m$:
  Nyström approximation $\boldsymbol{K} \approx \boldsymbol{K}_{nm}\boldsymbol{K}_{mm}^{-}\boldsymbol{K}_{mn}$,
  $\mathcal{O}(nmk + m^3)$ for a rank-k-approximate inverse of $\boldsymbol{K}_{mm}$.
- exploiting structure in $\boldsymbol{K}$ induced by the kernel: exact solutions but complicated maths, not applicable for all kernels.

... this is still an active area of research.